The Very Valuable Variable Value Functions

Howard Schreier
U.S. Dept. of Commerce

Abstract

Sometimes it is useful, within a DATA step, to inspect and utilize a variable's formatted value. In earlier versions of SAS® software, this was clumsy. Version 9 introduces the VVALUE function, which makes the process simpler. The related VVALUEX function makes the operation data-driven (rather than code-driven) and thus makes it possible to reference data by name at execution time.

VVALUE Function

The VVALUE function takes, as its only argument, the name of a variable or an array reference. It returns a character string containing the formatted value of the variable referenced.

Example: Format

```
proc format;
value myfmt (default=15)
            1 = 'Yes'
            2 = 'No'
            3 = 'Don''t Know'
            other = 'N/A'
            ;
run;
```


Example: Code and results

```
data demo;
retain a b c d e 3 ;
format a 4.1
       b 5.2
       d date9.
       e myfmt12.
       ;
put (_all_)(=) ;
run;
```

```
a=3.0 b=3.00 c=3
d=04JAN1960 e=Don't Know
```


Beyond the PUT statement

The PUT statement automatically applies the formats which have been associated with the variables.

But what if it's necessary to reference these formatted values within the DATA step? For example, one might want to test their content or embed them in longer strings.

PUT function

The PUT function is applicable in this situation.

However, it requires specification of a format. Why should that be mandatory when every variable has associated with it (either by earlier declaration or by default) a format?

Example: Requirement

Task: write the five variables to the log, one per line, with decimal points aligned. If there is no decimal point, the right alignment should be just clear of where the decimal point would be.

```
             3.0
             3.00
             3
      04JAN1960
     Don't Know
```


Example: Troublesome code

```
data _null_;
set demo;
put +11 +1 +1 a 4.1      -r;
put +10 +1 +2 b 5.2      -r;
put  +3       c best12.  -r;
put  +6       d date9.   -r;
put  +3       e myfmt12. -r;
run;
```


Using VVALUE: Code (1 of 2)

```
data _null_;
set demo;
array abcde (5) a b c d e ;
do i = 1 to 5;
 charvalue = vvalue(abcde(i) );
```


Using VVALUE: Code (2 of 2)

```
 indent = 16 -
  length(strip(charvalue));
 decimalpart =
  scan(charvalue,2,'.');
 if not missing(decimalpart)
  then indent = indent +
  length(decimalpart) + 1;
 put @indent charvalue;
 end;
run;
```


VVALUE workaround for 9.0 (1 of 2)

```
data _null_;
set demo;
charvalue = vvalue(a) ;
 link display ;
charvalue = vvalue(b) ;
 link display ;
charvalue = vvalue(c) ;
 link display ;
charvalue = vvalue(d) ;
 link display ;
charvalue = vvalue(e) ;
 link display ;
```

VVALUE workaround for 9.0 (2 of 2)

```
return;
display :
indent = 16 –
 length(strip(charvalue) ) ;
decimalpart =
 scan(charvalue,2,'.');
if not missing(decimalpart)
 then indent = indent +
 length(decimalpart) + 1;
put @indent charvalue;
return;
run;
```


VVALUEX Function

- Similar to the VVALUE function in terms of what it does
- Much different in that it requires as its argument a character expression which evaluates to the name of a variable
- Effectively equivalent expressions:

```
vvaluex("myvar")
vvalue(myvar)
```


VVALUEX: Code for example (1 of 2)

```
data _null_;
if _n_=1 then set demo;
infile cards;
input chosenvar $ ;
charvalue = vvaluex(chosenvar) ;
indent = 16 –
 length(strip(charvalue) );
decimalpart =
 scan(charvalue,2,'.');
if not missing(decimalpart)
 then indent = indent +
 length(decimalpart) + 1;
```


VVALUEX: Code for example (2 of 2)

```
put @indent charvalue;
cards;
c
e
a
d
;
```


VVALUEX: Example results

```
          3
Don't Know
          3.0
 04JAN1960
```


VVALUEX: Beyond the example

This example of VVALUEX usage was designed to parallel the example used for VVALUE, but that actually hides much of the versatility of VVALUEX. What we actually have here is a direct way to reference data items, at execution time, by name.

Conclusion

The VVALUE function makes it possible for SAS code to reference the formatted value of a variable without having to specify or retrieve the variable's format. The VVALUEX function goes a step further by allowing the process to be data-driven rather than code-driven.

About the Speaker

Howard Schreier

U.S. Dept. of Commerce
Washington DC 20230
USA

(202) 482-4180

Howard_Schreier@ita.doc.gov
http://howles.com/sugi/